

FIG. 1

Info records – Contain information about the following:

AssemblyInfo = Full name of an assembly that when loaded contains Types used in the construction of an object tree

TypeInfo = full name of the type of an object, with a reference to the Assembly record where it is defined

AttributeInfo = Full name of an attribute, or property, on an object, with a reference to the Type record that defines this attribute, and the type record that defines the type of the property.

FIG. 2

Structure records – Contain information about the object tree structure, and make references to the info records for specific types and attributes. A full representative set of these records are:

Unknown = Placeholder for unknown record

StartDocument = Start of a XAML or BAML document

EndDocument = End of a XAML or BAML document

Element = Start of a DependencyObject

EndElement = End of a Dependency Object

ParseLiteralContent = A section of literal content that is interpreted by an object that implements the IparseLiteralContent interface

XmlnsProperty = Information about an xmlns declaration encountered in a xaml file

DependencyProperty = An attached DependencyProperty associated with a DependencyObject

RoutedEvent = Information about an Avalon RoutedEvent associated with a DependencyObject

Text = Text content under a DependencyObject or a CLR object

ComplexDependencyProperty = The start of a DependencyProperty that is represented as an element subtree

EndComplexDependencyProperty = The end of a DependencyProperty that is represented as an element subtree

ClrObject = The start of a CLR object, which is an object that does not derive from DependencyObject

EndClrObject = The end of a CLR object

ClrProperty = A property on a CLR object

ClrArrayProperty = The start of an array property on a CLR object

EndClrArrayProperty = The end of an array property on a CLR object

ICollectionProperty = The start of a property on an object that implements the ICollection interface

EndICollectionProperty = The end of an ICollection property

ClrComplexProperty = The start of a property on a CLR object that is represented by a tree of objects

EndClrComplexProperty = The end of a CLR complex property

IncludeTag = A reference to include another section of BAML in place of this record

IDictionaryProperty = The start of a property on an object that implements IDictionary.

EndIDictionaryProperty = The end of a property that implements IDictionary

DictionaryKeyTag = An attribute that defines a key used for placing the parent object into an IDictionary

DependencyPropertyCustom = A DependencyObject that has its own custom binary representation

PIMapping = A mapping of assembly to CLR and XML namespace, used for type loading and resolution

ClrPropertyCustom = A CLR property that has its own custom binary representation

FIG. 3

StartDocument	
Version = 1.2	
AssemblyInfo	- ID = 1 Name = "PresentationFramework"
TypeInfo	- ID = 1 Name = "System.Windows.Documents.DockPanel" AssemblyID = 1
StartElement	- TypeID = 1
Xmlns	- Prefix = "" Value = "http://schemas.microsoft.com/2003/xaml"
TypeInfo	- ID = 2 Name = "System.Windows.Documents.Button" AssemblyID = 1
StartElement	- TypeID = 2
AssemblyInfo	- ID = 2 Name = "PresentationCore"
TypeInfo	- ID = 2 Name = "System.Windows.Media.Brush" AssemblyID = 2
TypeInfo	- ID = 3 Name = "System.Windows.Controls.Panel" AssemblyID = 1
AttributeInfo	- ID = 1 Name = "BackgroundProperty" OwnerType = 3 PropertyType = 2
DependencyProperty	- AttributeID = 1 Value = "Blue"
EndElement	
EndElement	
EndDocument	

FIG. 4

```

///<summary>
/// Serialize this object using the passed writer
///</summary>
/// <remarks>
/// This is called ONLY from the Parser and is not a general public method.
/// </remarks>
//
// Format of serialized data:
// first byte  other bytes  format
// 0AAAAAAA  none      Amount from 0 to 127 in AAAAAAA, Pixel UnitType
// 100XXUUU  one byte   Amount in byte - 0 - 255, UnitType in UUU
// 110XXUUU  two bytes  Amount in int16 , UnitType in UUU
// 101XXUUU  four bytes Amount in int32 , UnitType in UUU
// 111XXUUU  eight bytes Amount in double, UnitType in UUU
//
public void IBamlSerialize.SerializeOn(BinaryWriter writer, string stringValue)
{
    byte unitAndFlags = (Byte)UnitType;
    int intAmount = (int)Value;
    if ((float)intAmount == Value)
    {
        if (intAmount <= 127 &&
            intAmount >= 0 &&          // 0 - 127 and Pixel
            UnitType == UnitType.Pixel)
        {
            writer.Write((byte)intAmount);
        }
        else if (intAmount <= 255 &&
            intAmount >= 0)          // Unsigned byte
        {
            writer.Write((byte)(0x80 | unitAndFlags));
            writer.Write((byte)intAmount);
        }
        else if (intAmount <= 32767 &&
            intAmount >= -32768)    // Signed Short Integer
        {
            writer.Write((byte)(0xC0 | unitAndFlags));
            writer.Write((Int16)intAmount);
        }
        else                          // Signed Integer
        {
            writer.Write((byte)(0xA0 | unitAndFlags));
            writer.Write(intAmount);
        }
    }
    else                          // Double
    {
        writer.Write((byte)(0xE0 | unitAndFlags));
        writer.Write((Double)Value);
    }
}

```

FIG. 5

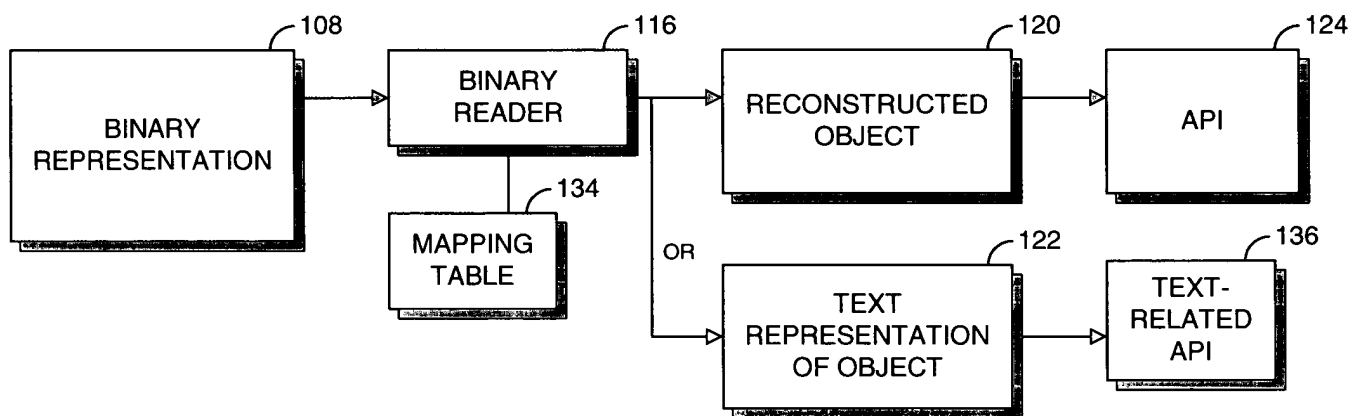


FIG. 6

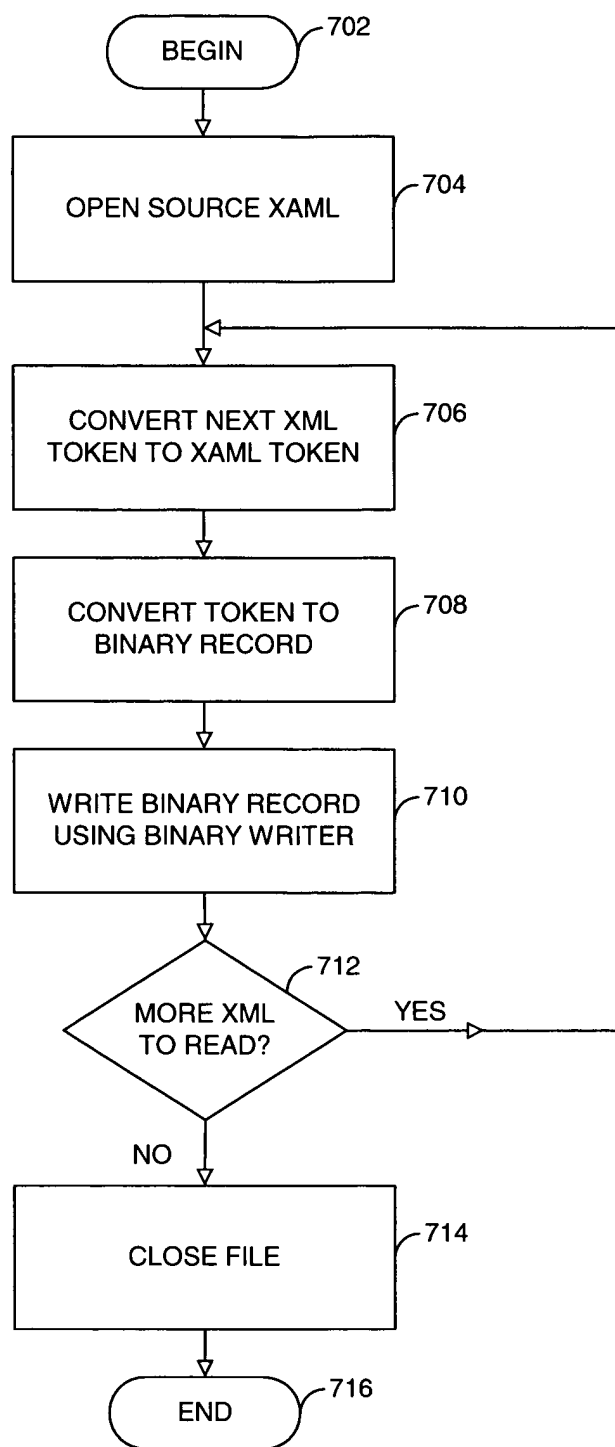


FIG. 7

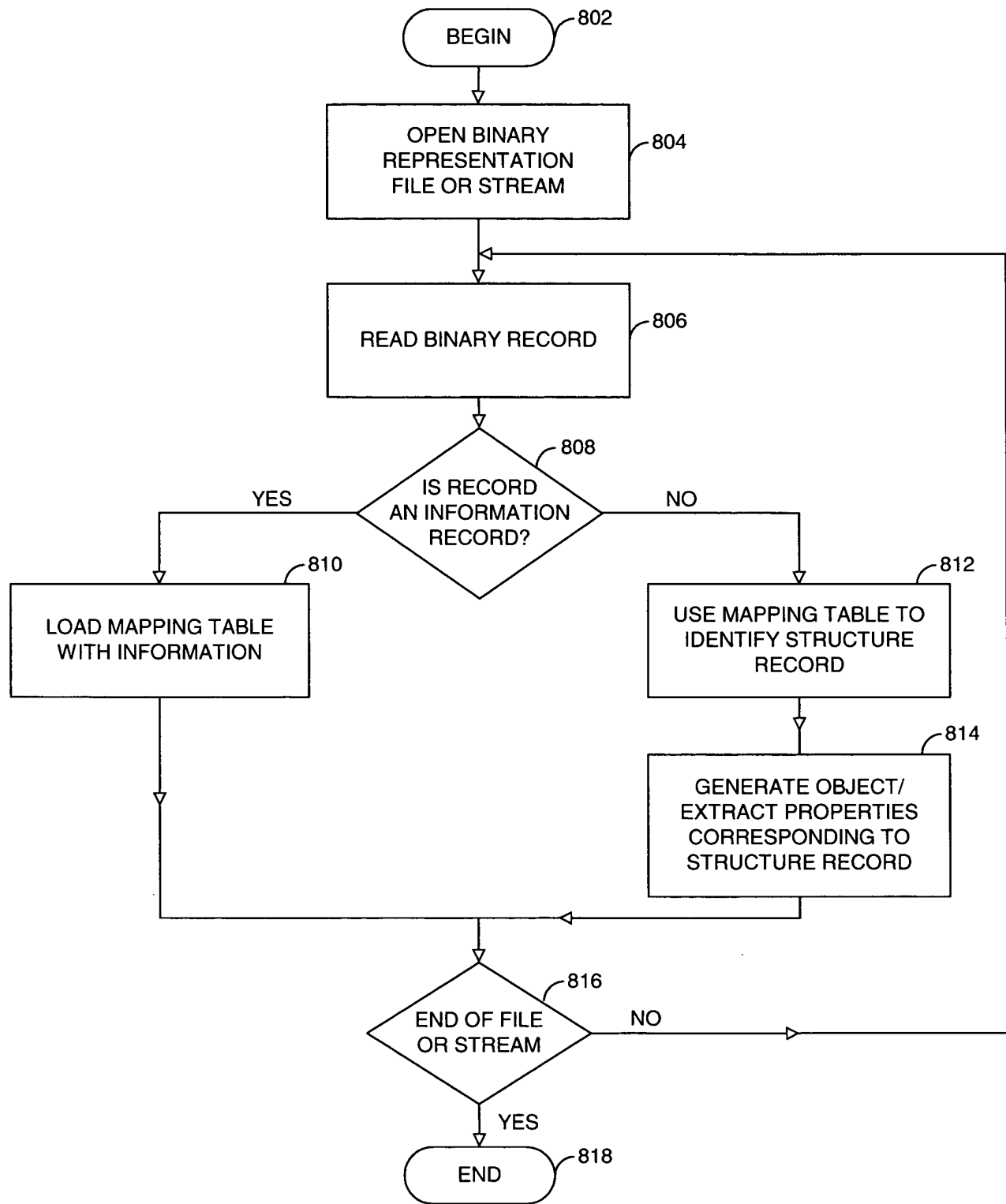


FIG. 8